Video Markup Language (VideoML) – An XML-based Markup Language for Video Annotation and Analysis

Honglin Li, Yi Zhao, and Stanley C. Ahalt
Department of Electrical Engineering
The Ohio State University

ABSTRACT

It is well established that the multimedia community, including both producers and the consumers of multimedia data, needs efficient mechanisms to describe, organize, store, query, and access collected video data. While various techniques have been proposed to extract information from raw video data, there is still a great deal of discussion and research centered on how to organize extracted data as well all other meta-data associated with digital video. Video meta-data includes information such as camera, lighting, and geographical information, as well as information derived from the video such as content and organization. What is needed is a description method that supports the both descriptors and the organization of the video information so the meta-data is easy to store, access, exchange, transmit, and process. The description should be hierarchical in order to represent the various temporal resolution levels of video, including the raw video, story-units, scenes, shots, and frames. Further, the description should be self-descriptive so that arbitrary applications can determine what the metadata means. And the description should be extensible since more information may need to be added as the video is processed or utilized for different functions. Finally, the description should readily inter-operate with other applications. In this paper, we propose Video Markup Language (VideoML) – an XML-based makeup language– to organize the video meta-data for video annotation and analysis applications. Considering the popularity of XML and the emergent MPEG-7 standard, we anticipate that VideoML could play an important role in video data description and exchange for a number of video annotation and analysis applications, including those applications that are of particular interest to the DoD.

1. INTRODUCTION

In many scientific, engineering, industrial, and military applications huge amounts of data are generated on a daily basis. For example, data may be collected in the course of as part of experiments conducted at geographically distributed locations at different times. Additionally, in some cases it is not practical to consolidate all and house the data on a single data server or database. Finally, in many cases the structure of the data is complex and typically heterogeneous. Nevertheless, in many collaborative research environments, scientists and engineers would like to transmit queries and obtain data useful for their applications by using the web. Thus we need to find a method of efficiently describing, organizing, storing, querying, and accessing the data that has been collected. When the data consists largely of video, the difficulties are even greater because the data is not readily organized.

The past decade has witnessed significant advances in multimedia technology. Increasing computing power, broadband networks, new compression techniques, cheaper storage, and inexpensive and ubiquitous acquisition devices make multimedia data − image, video, audio, speech, and graphics− virtually omnipresent. Through VCD and DVD players, MP3 players, and the Internet, everyone is becoming a multimedia consumer. Moreover, because of the popularity of digital cameras and camcorders everyone is becoming a potential multimedia content producer, capable of publishing the multimedia data using the web. Add to this mixture emerging multimedia data sources such as digital libraries, distance learning, telemedicine, and video surveillance, and one can see that there is an explosion of multimedia data occurring – and which will continue to grow for the foreseeable future. It is a reality that the sheer volume of digital multimedia data that is being generated worldwide threatens to overwhelm multimedia consumers. Indeed, it can be argued that – without efficient means to access this vast data resource – our global society may be overwhelmed and never make appropriate use of this unprecedented resource. Thus we argue that there is a growing need to investigate how we access multimedia data. In this paper, we focus on the most challenging part of multimedia data, namely, video data. Unfortunately the current tools used to organize and access video data are relatively primitive compared to the tools that have been developed over the years for textual data.

In recent years, active research has been conducted in order to provide methodologies to access video data. Various techniques, such as video indexing and video annotation, have been proposed to extract information from the raw video data. Video meta-data, the data that describes the video content, is one form of information extracted from the raw video data. Meta-data can be text-based or symbolic, derived at low-levels of abstraction or higher-levels of abstraction, and the data can be automatically generated or manually annotated. Video access will, in many cases, be conducted solely based on video meta-data. The video metadata can consist of various video features such as spatial features, temporal features, bibliographical features, structural features, and grammar symbols, and consequently the organization of the video metadata is problematic. What is needed is a description method that supports both descriptors and the organization of the video information so that it is easy to store, access, exchange, transmit, and process. The description should be hierarchical in order to represent the various temporal resolution levels of video, including the raw video, story-units, scenes, shots, and frames. Further, the description should be self-descriptive so that arbitrary applications can determine what the metadata means. The description should be extensible since more information may need to be added as the video is processed or utilized for different functions. Finally, the description should readily inter-operate with other applications.

Current research is underway in an attempt to provide an appropriate description technique that satisfies these requirements. The Synchronized Multimedia Integration Language (SMIL, pronounced "smile") [1] enables simple authoring of interactive audiovisual presentations. SMIL is typically used for "rich media"/multimedia presentations which integrate streaming audio and video with images, text or any other media type. SMIL, however, does not solve the problem of how to organize video meta-data. The Multimedia Retrieval Markup Language (MRML), formally specified in [2], provides a framework that separates query formulation from actual query shipping. MRML is designed to markup multi-paradigm queries for multimedia databases. Researchers [3] have demonstrated a video annotation engine in which the video meta-data is organized using SGML (Standard Generalized Markup Language). Finally, MPEG-7 [4, 5], formally called the Multimedia Content Description Interface, specifies the use of XML (eXtensible Markup Language) [6] as the language for the Description Definition Language (DDL) used to organize MPEG7 video meta-data. For example, MPEG-7 specifies a collection of standard video descriptors –representations of video features– covering color, motion, shape and texture. However, MPEG-7 provides only a framework, and specific markup languages need to be designed for specific application areas.

In this paper we introduce a specification we refer to as the Video Markup Language (VideoML) that is designed for video annotation and analysis applications.

2. XML BASICS

XML stands for eXtensible Markup Language, which is a markup language much like HTML. However, there are some fundamental differences between XML and HTML. Here we list some key characteristics of XML in order to explain why we choose to use XML as the language we employ to organize video meta-data.

- XML was designed to describe data by focusing on what the data consists of. In contrast, HTML was designed to display data and focuses on how the data appears. Thus, while HTML is focused on information display, XML is focused on information description. XML is not a replacement for HTML, but a complement to HTML.
- XML tags are not predefined in XML. You can define your own tags in XML, unlike HTML tags that are predefined.
- XML focuses on the data structure.
- XML is inherently hierarchical.
- With DTDs (Document Type Definitions), an XML document is totally self-descriptive and can be validated against the specified DTD.
- XML is extensible. People are free to define their own tags.
- XML is neutral with regard to databases. Increasingly more databases support XML.
- XML is growing in popularity, with more and more applications supporting XML.

A DTD (Document Type Definition) defines the legal elements of an XML document. The purpose of a DTD is to define the legal building blocks of an XML document. A DTD defines the document structure with a list of legal elements. A "well formed" XML document should conform to the rules of a DTD. By defining new elements in a DTD, one can use customized tags, which makes XML extensible. Of course, different applications need different sets of tags and consequently need different DTDs.

DTDs are easy to use, but have limited descriptive power. To address this problem the W3C has launched an effort to develop a replacement called XML Schema. Schemas support more modern modeling concepts, they are better suited for data exchange and application integration, and they are written as XML documents. The key task of implementing this framework is to design the appropriate XML models –the Document Type Definition (DTD) or Schema in XML terminology.

3. APPLICATION FRAMEWORK

3.1 General Framework

Figure 1 illustrates a general framework of how data can be organized, stored, and accessed. This framework is applicable to many scientific and engineering applications. The data cluster consists of nodes that store the data. As previously mentioned, these data nodes might be geographically scattered. The data on these data nodes may be organized using XML and stored as flat files or databases. The data cluster also has one node we refer to as the data server, which essentially records the data directory. The data directory itself may also be organized using XML. The computing cluster consists of nodes that provide computing power. Through the coordination of the compute server, data-intensive computing tasks can be distributed and performed in parallel. The coordinating instructions necessary for these activities may also be organized using XML. The application server is responsible for accepting requests from the client, and then sending data query request to data server. The data server locates the data, which might come from several data nodes, and sends the data back to the application server. After obtaining the required data, the application server sends a computing request to the compute server with the data. The compute server subsequently designs the computing strategy and distributes the computing task among the computing cluster resources, and finally sends the results back to application server. The application server then sends the result back to the client. In this process, all the following elements could be organized using XML.

- The request from client to the application server.
- The data query request from the application server to the data server.
- The data query result.
- The computing request from the application server to the compute server.
- The coordinating instructions issued by the compute server.
- The final result from the compute server to the application server.
- The final results from the application server to the client.

Obviously this general framework is centered on the use of XML.

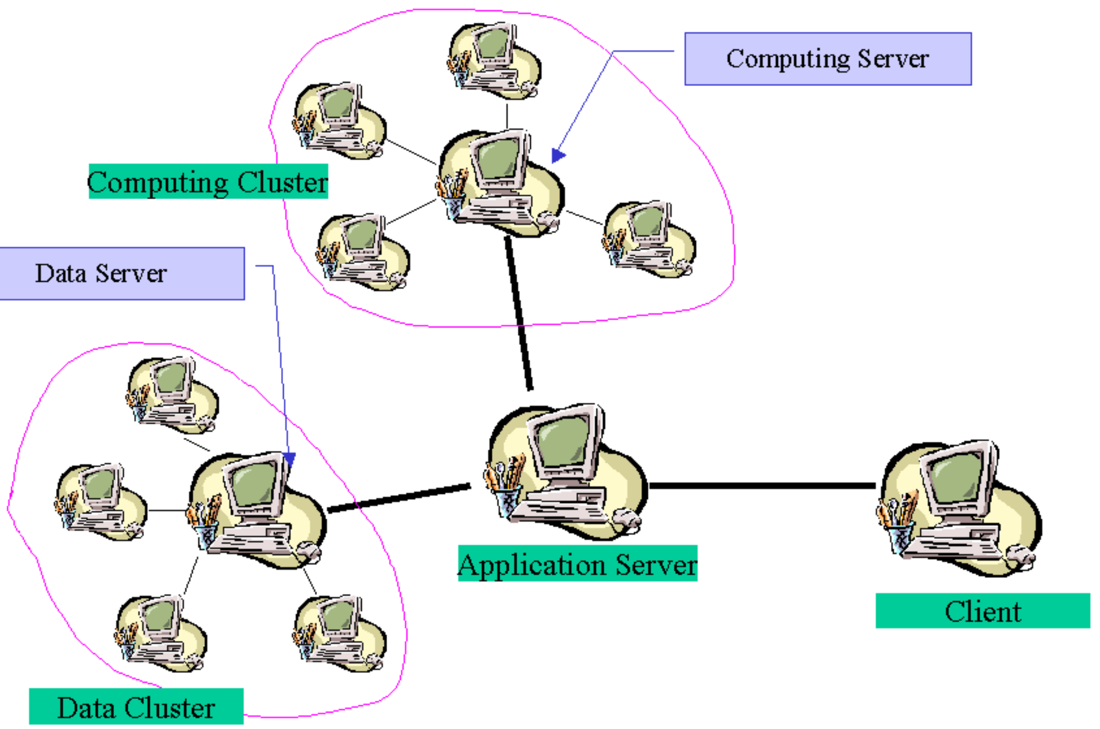


Figure 1. The general framework.

3.2 Application Framework on Video Annotation and Analysis

One possible system architecture is shown in Figure 2. The raw video data first undergoes a video segmentation process, which in this case is a shot detection operation that divides the video data into basic units –shots. Information extraction then operates on the shots in order to produce video meta-data that describes the video content. Next the video meta-data is organized using XML. Various video operations can subsequently be performed on the video meta-data.

Shot Detection

The structure within a video stems from the fact that video streams are formed by editing different video segments known as shots. A shot is a sequence of frames generated during a continuous camera operation and represents continuous action in time and space. Shots can be joined together in either an abrupt transition mode, in which two shots are simply concatenated, or through gradual transitions, in which additional frames may be introduced using editing operations such as dissolve, fade-in, fade-out, and wipe. The purpose of shot detection is to detect the shot boundaries, either abrupt transition or gradual transition, and so segment the video stream into shots, which are the basic video units for subsequent processing.

There are a number of shot detection algorithms [7-23] that fall into two categories: those that work in the uncompressed domain and those that operate in the compressed domain.

Information Extraction

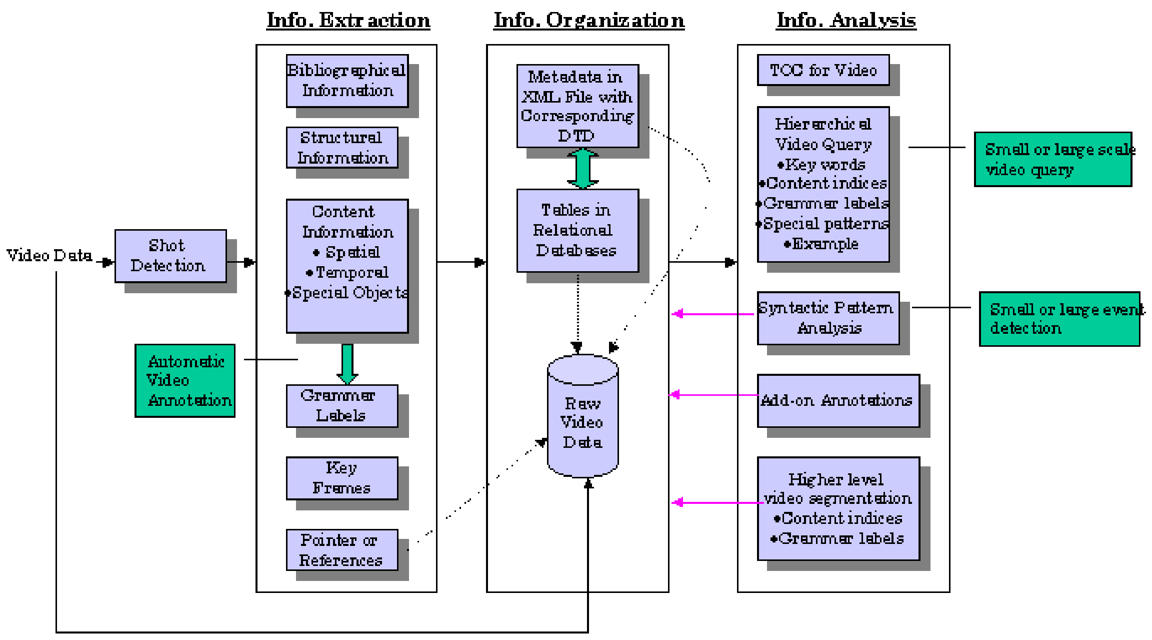

Info. Extraction
Info. Organization
Info. Analysis
Video Data
Shot Detection
Automatic Video Annotation
Bibliographical Information
Structural Information
Content Information • Spatial • Temporal •Special Objects
Grammar Labels
Key Frames
Pointer or References
Metadata in XML File with Corresponding DTD
Tables in Relational Databases
Raw Video Data
TOC for Video
Hierarchical Video Query •Key words •Content indices •Grammar labels •Special patterns •Example
Syntactic Pattern Analysis
Add-on Annotations
Higher level video segmentation •Content indices •Grammar labels
Small or large scale video query
Small or large event detection

Figure 2. Proposed system architecture.

After the video is segmented into shots the video content can be extracted from the shots in the form of various video features. By analyzing the shot level video features, one can group shots into high-level units – scenes. Additionally, one can go further and group scenes into story-units that compose the whole video. Video features can have various properties such as the abstraction level, the temporal resolution level [24], and the aggregation level [24]. Moreover, video features can either be manual or automatic.

Associated with any video data are numerous features, from among which we choose the following categories of features for our proposed system.

- *Bibliographical information.* Bibliographical information is textual information that captures the production source, the abstract, the frame rate, the video resolution, the compression format, the category, the frame numbers, and so forth. This information is most frequently annotated manually, and mainly for high temporal level units.

- *Structural information*. Structural information encodes the hierarchical structure of the video, in which a whole video is divided into many story-units that in turn are divided into scenes, shots, and frames. This structural information can readily be converted to a TOC of the video that expedites the video browsing process.
- *Low-level content-based information*. Low-level content-based information consists of various features that fall into two basic categories: spatial, and temporal. Spatial features refer to features such as color, texture, and shape; temporal features refer to features such as object motion and camera motion that can only be collected over multiple frames.
- *Grammar labels***.** Video shots can be classified based on low-level video features and one or more (aggregate) labels can then be assigned to each shot. We refer to the aggregate labels as grammar labels since they can be potentially used in syntactic video analysis. In most cases a predictive model must be constructed based on the low-level video features in order to generate the grammar labels. This process can also be called automatic video annotation, and represents an integral part of this proposal.
- *Key frames extraction.* A set of key frames can be extracted from each shot in order to a) represent the shot and make the presentation of the shot easier and b) extract spatial features from the key frames instead of the entire collection of frames that comprise the shot. A number of algorithms [25-28] have been proposed for key frame extraction, typically based on visual differences or on motion patterns.
- *Pointers and references.* In order to locate the raw video data that corresponds to a particular portion of video meta-data, pointers are added. For example, the starting frame number and the ending frame number of shots. We would sometimes like to know where relevant or similar video data units (shots or scenes) to another video unit are, and references are used to refer to the relevant or similar video data units. Through the use of pointers and references, the video meta-data can be stored separately from the raw video data without losing necessary links.

After these, or similar, video features are extracted they can be organized into video meta-data. To organize this meta-data, we have designed VideoML – an XML-based video markup language for video annotation and analysis.

Information Analysis

Once we have the available video information and the meta-data organized in a convenient format there are various video analysis operations that can be performed. Analysis results can be appended to the video meta-data. Several important applications of video meta-data are identified below.

Table of contents (TOCs)

Like a TOC for a book, a video TOC is an effective tool for assisting people in accessing video information. Because of the hierarchical structure of the video meta-data organized using VideoML, as well as the pointers associated with each temporal resolution level, a TOC is naturally and directly drawn from the meta-data.

Hierarchical video query

Due to the extensive information contained in the video meta-data, various video queries can be applied to the video data. For example, we can perform query by key words, query by low-level features, query by grammar labels, query by special patterns, and query by example. Nevertheless, the hierarchical structure of the video meta-data enables us to retrieve video according to different time scales.

Syntactic pattern analysis

The grammar labels we propose to use provide the possibility that syntactic pattern analysis can be performed on the video meta-data. Syntactic pattern analysis is an appropriate tool for analyzing structured pattern, which cannot be easily accomplished by traditional statistical pattern analysis methods. Many video patterns are structured, and this makes syntactic pattern analysis a good choice for video pattern analysis.

4. VIDEOML – VIDEO MARKUP LANGUAGE

VideoML is designed to facilitate the process of video annotation and video analysis. By employing VideoML, the heterogeneous video information extracted from the raw video data can be effectively organized, and consequently easily exchanged among different applications. The ongoing work will extend VideoML to provide descriptive power for structuring the communication information among clients and servers. Furthermore, VideoML will incorporate components that deal with the computational effort in video annotation and analysis applications.

4.1 Design of VideoML

Designing an XML model is equivalent to prepare the corresponding DTD or XML Schema. The syntax of DTD and XML schema is very complex, and thus it is a challenging task to directly write the DTD or XML Schema. However, there are XML modeling tools available in the market, and these tools make the XML design significantly easier. TurboXML [29] is one good XML modeling tool and it has been used to design VideoML. By using TurboXML, the designer does not need to be concerned with the syntax of DTD and XML Schema. The design work is conducted in an intuitive and graphical way, and the corresponding DTD or XML Schema is generated by saving the model to a disk file. Figure 3 is a screenshot of TurboXML.

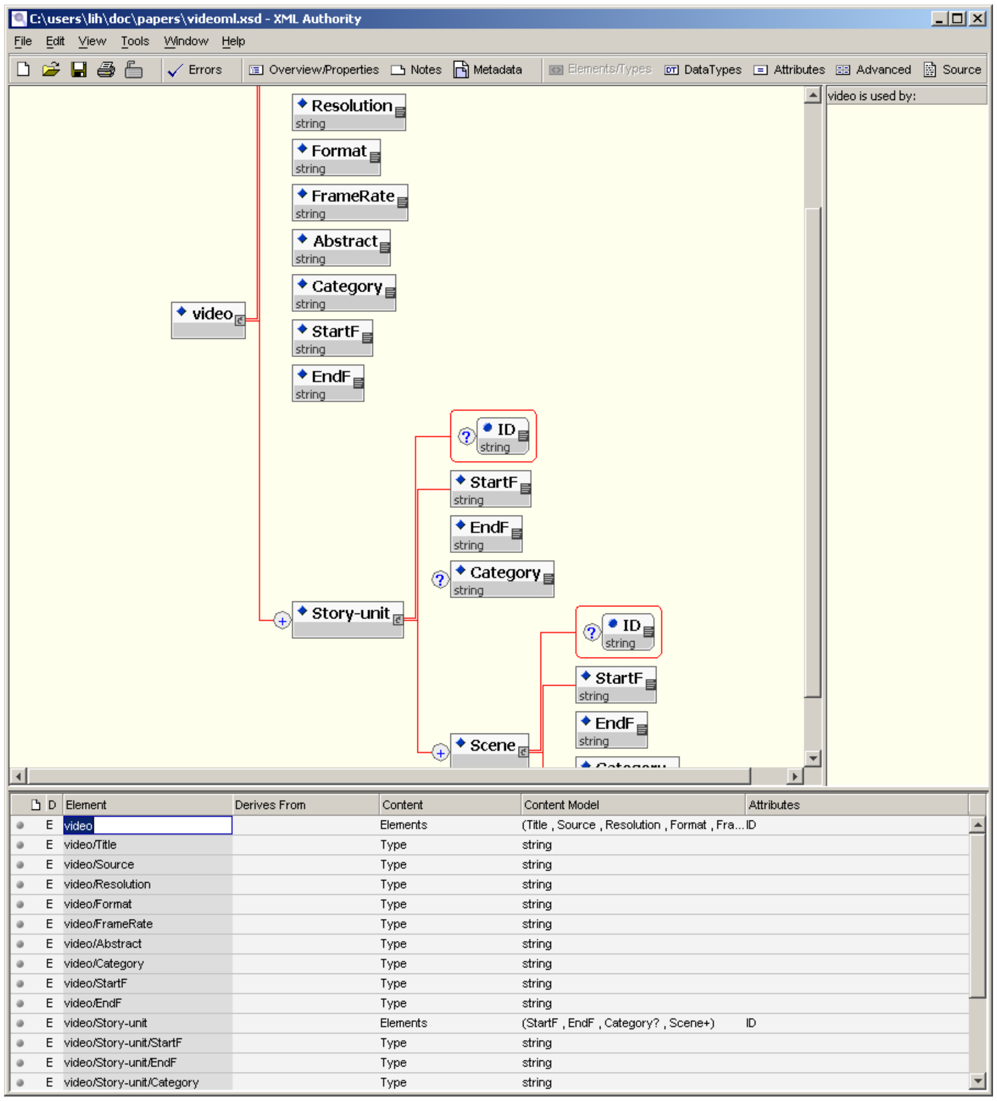
4.2 Key Features of VideoML

The formal specification of VidoeML, or either the DTD or XML Schema for VideoML, is not fully described here because it is difficult to present the complete schema in a compact fashion, and the schema is not necessarily as informative as might be desired. However, for informational purposes a few of the key features of VideoML are identified in this section. Currently, VideoML supports features including:

- *Bibliographical information.* The bibliographical information is usually associated with high level video units such as the whole video or the story-units. The VideoML code for the bibliographical information for a video has the form:

```
<Video>
       <Title> NBA Final </Title>
       <Source> CNN </Source>
       <Resolution> CIF(352x288) </Resolution>
       <Format> MPEG-1 </Format>
       <FrameRate> 30 </FrameRate>
       <Abstract>
          The first game in NBA final stage between LAKERS and
        76ers </Abstract>
       <Category> Sports </Category>
</Video>
```

- *Structural information.* VideoML provides constructs to describe the hierarchical structure of each video. That is, the various level of video segmentation can be easily discerned from a VideoML file. This is achieved by utilizing the inherent hierarchical structure of XML. For



D	Element	Derives From	Content	Content Model	Attributes
E	video		Elements	(Title , Source , Resolution , Format , Fra...	ID
E	video/Title		Type	string	
E	video/Source		Type	string	
E	video/Resolution		Type	string	
E	video/Format		Type	string	
E	video/FrameRate		Type	string	
E	video/Abstract		Type	string	
E	video/Category		Type	string	
E	video/StartF		Type	string	
E	video/EndF		Type	string	
E	video/Story-unit		Elements	(StartF , EndF , Category? , Scene+)	ID
E	video/Story-unit/StartF		Type	string	
E	video/Story-unit/EndF		Type	string	
E	video/Story-unit/Category		Type	string	

Figure 3. Using TurboXML to design an XML model.

example, the following code segment describes the structural information of one video sequence:

```
<Video>
   <!-- Video level features -->
   <Story-unit ID="001">
      <!-- Story-unit level features -->

      <Scene ID="001">
         <!-- Scene level features -->
         <Shot ID="001">
            <!-- Shot level-->
         </Shot>
      </Scene>
   </Story-unit>
 </Video>
```

- *Low-level, content-based video features.* VideoML code constructs can describe spatial features like color histograms, edge orientation histograms, and temporal features like differences between color histograms and motion vectors.

```
<SpatialFeatures>
   <ColorHistogram> </ColorHistogram>
   <EdgeOrientHist> </EdgeOrientHist>
</SpatialFeatures>
<TemporalFeatures>
   <MotionVectors> </MotionVectors>
   <DiffColorHist> </DiffColorHist>
</TemporalFeatures>
```

- *Grammar labels or automatic annotation results.* For example, `<Label> B </Label>`, embedded in a shot element, identifies that the shot has a label B. This label might be generated from an automatic annotation module and it is useful for many subsequent operations such as syntactic video analysis.
- *Pointer information.* The element `<StartF>` and `<EndF>` denotes the starting frame number and the ending frame number for specific video units, which can be story-units, scenes, or shots.
- *Key frames.* The element `<R-frame>,` embedding in a shot element, denotes the key frame number(s) for the shot.
- *Communication information among the various nodes of a network such as clients, data servers, and computing servers.* This is an area of ongoing research in which we are attempting to incorporate data in the video meta-data that can be used to describe the distributed data constituting the data.
- *Computational instructions.* Here we use computational instructions that may be useful for distributed and parallel analysis of the video data. Again, this is an area of ongoing research.

4. 3 An Example VideoML File

In order to give readers a flavor of VideoML, Figure 4 shows one example VideoML file.

```
<?xml version="1.0"?>
<VideoML>
<Video ID = "001">
    <Title> NBA Final </Title>
    <Source> CNN </Source>
    <Resolution> CIF(352x288) </Resolution>
    <Format> MPEG-1 </Format>
    <FrameRate> 30 </FrameRate>
    <Abstract> The first game in NBA final stage between LAKE and 76ers
</Abstract>
    <Category> Sports </Category>
    <StartF> 1 </StartF>
    <EndF> 16,000 </EndF>
    <Story-unit ID = "001">
        <StartF> 1 </StartF>
        <EndF> 1012 </EndF>
        <Category> Actions </Category>
        <Scene ID = "001" >
            <StartF> 1 </StartF>
            <EndF> 300 </EndF>
            <Category> Lake Offense </Category>
            <Shot ID = "001">
                <StartF> 1 </StartF>
                <EndF> 98 </EndF>
```

```
                    <R-Frame> 54 </R-Frame>
                    <SpatialFeatures>
                        <ColorHistogram> 25,39,34,30 </ColorHistogram>
                        <EdgeOrientHist> 34,34, 5, 54 </EdgeOrientHist>
                    </SpatialFeatures>
                    <TemporalFeatures>
                        <MotionVectors> 34,45, 54 ,34 </MotionVectors>
                        <DiffColorHist> 3,4,5, 3 </DiffColorHist>
                    </TemporalFeatures>
                    <AudioFeatures>
                        <Pitch> 34 </Pitch>
                    </AudioFeatures>
                    <ClosedCaptions>
                        <HostA> Ivanson is flying! </HostA>
                        <HostB> I bet he will be the MVP this year!</HostB>
                    </ClosedCaptions>
                    <Label> B </Label>
                </Shot>
                <Shot ID = "003">
                </Shot>
                <Shot ID = "004">
                </Shot>
            </Scene>
            <Scene ID = "002">
            </Scene>
            <Scene ID = "003">
            </Scene>
    </Story-unit>
    <Story-unit ID = "002">
        <StartF> 1013 </StartF>
        <EndF> 2056 </EndF>
        <Category> Commercials </Category>
    </Story-unit>
    <Story-unit ID = "003">
    </Story-unit>
    <Story-unit ID = "004">
    </Story-unit>
</Video>
</VideoML>
```

Figure 4. Example VideoML file.

5. CONCLUSIONS

In this paper, we have briefly presented the VideoML (Video Markup Language) for video annotation and analysis applications. VideoML focuses on the description and organization of video meta-data. We are still in the process of designing additional modules to describe client service requests, data query requests, computing requests, and the results format. VideoML is particularly timely in that while MRML provides some functionality for structuring video queries, it is not generally suited for video annotation and analysis applications. We also plan to extend VideoML so that it not only organizes various video features within a flexible structure, but it also provides components that deal with the computational aspects of processing either the video data or the descriptors. For example, a VideoML file could readily specify recommendations for parallel computation, optimization, and debugging. One possibility that is being explored is the use of VideoML with MatlabMPI, a Matlab implementation of the Message Passing Interface (MPI).

REFERENCES

1. http://www.w3.org/TR/smil20.
2. Muller, W., et al., *MRML: Towards an extensible standard for multimedia querying and benchmarking*. 1999, University of Geneva: Genève, Switzerland.
3. Carrer, M., et al., *An annotation engine for supporting video database population*. Multimedia Tools and Techniques, 1997. **5**: p. 233-258.
4. Pereira, F. and R.H. Koenen, *MPEG-7: status and directions*, in *Multimedia Systems, Standards, and Networks*, A. Puri and T. Chen, Editors. 2000, Marcel Dekker, Inc.: New York.
5. Martinez, J.M., *Overview of the MPEG-7 standard*. 2001, ISO/IEC JTC1/SC29/WG11.
6. http://www.w3.org/XML/.
7. Kasturi, R. and R. Jain. *Dynamic vision*. in *Computer Vision: Principles*. 1990. Los Alamitos, CA: IEEE Computer Society.
8. Nagasaka, A. and Y. Tanaka. *Automatic video indexing and full video search for object appearance*. in *IFIP: Visual Database Systems II*. 1992.
9. Gargi, U., et al. *Evaluation of video sequence indexing and hierarchical video indexing*. in *Proceedings of SPIE: Storage Retrieval for Image and Video Databases III*. 1995.
10. Tonomura, Y. *Video handling based on structured information for hypermedia systems*. in *ACM Proceedings: International Conference on Multimedia Systems*. 1991.
11. Shahraray, S., *Scene change detection and content-based sampling of video sequences*, in *Digital Video Compression: Algorithms Tech. 2419*. 1995. p. 2-13.
12. Swanberg, D., C.-F. Shu, and R. Jain. *Knowledge guided parsing in video databases*. in *Proc. SPIE 1908*. 1993.
13. Zhang, H.J., et al., *Automatic partitioning of full motion video*. ACM Multimedia Systems, 1993. **1**(1): p. 10-28.
14. Hampapur, A. and R. Jain, *Production model based digital video segmentation*. Multimedia Tools and Applications, 1995. **1**: p. 9-46.
15. Aigrain, P. and P. Joly, *The automatic real-time analysis of film editing and transition effects and its applications*. Comput. Graphics, 1994. **18**(1): p. 93-103.
16. Little, T.D.C., et al. *A digital on-demand video service supporting content-based queries*. in *Proc. ACM Multimedia*. 1993. Anaheim, CA.
17. Zhang, H.J., et al. *Video parsing compressed data*. in *Proc. SPIE: Image Video Processing II*. 1994.
18. Zhang, H.J., et al., *Video parsing and browsing using compressed data*. Multimedia Tools Applications, 1995. **1**: p. 89-111.
19. Arman, F., A. Hsu, and M.-Y. Chiu. *Image processing on compressed data for large video databases*. in *Proc. SPIE: Storage Retrieval Image Video Databases*. 1993.
20. Arman, F., A. Hsu, and M.-Y. Chiu. *Feature management for large video databases*. in *ACM Multimedia 93*. 1993.
21. Liu, X., C.B. Owen, and F. Makedon, *Automatic video pause detection filter*. 1997, Dartmouth College, Computer Science: Hanover, NH.
22. Liu, H.-C.H. and G.L. Zick. *Scene decomposition of MPEG compressed video*. in *Digital Video Compression: Algorithms Tech*. 1995.
23. Lee, J. and B.W. Dickinson. *Multiresolution video indexing for subband coded video databases*. in *Proc. SPIE: Image Video Process. II*. 1994.
24. Lienhart, R., W. Effelsberg, and R. Jain, *VisualGREP: A systematic method to compare and retrieve video sequences*. Multimedia Tools and Applications, 2000. **10**: p. 47-72.
25. Ueda, H., T. Miyataka, and S. Yoshizawa. *IMPACT: An interactive natural-motion-picture dedicated multimedia authoring system*. in *Proc. Human Factors in Computing Systems CHI'91*. 1991.
26. Wolf, W. *Key frame selection by motion analysis*. in *Proc. of IASSP'96*. 1996.
27. Yeung, M.M. and B. Liu, *Efficient matching and clustering of video shots*. 1995, Princeton University.
28. Xiong, W., J.C.M. Lee, and R.H. Ma, *Automatic video data structuring through shot partitioning and key-frame computing*. Machine Vision and Applications, 1997. **10**: p. 51-65.
29. http://www.tibco.com/products/extensibility/solutions/turbo_xml.html.